



(11) (A) No. **1 187 197**

(45) ISSUED 850514

(52) CLASS 354-237

(51) INT. CL. G06F 13/00<sup>3</sup>

(19) (CA) **CANADIAN PATENT** (12)

(54) Segmented Storage Logging and Controlling

(72) Douglas, Gavin L.,  
U.S.A.

(73) Granted to International Business Machines Corporation  
U.S.A.

(21) APPLICATION No. 291,920

(22) FILED 771129

(30) PRIORITY DATE U.S.A. (762,379) 770125

No. OF CLAIMS 5

**Canada**

MAY 14 1985

1187197

A system and method are disclosed for controlling text storage in a text processing system using a segmented, serial bulk storage device. A plurality of storage segments of equal length are further subdivided into portions of equal length. A typical page of text occupies more than one portion of a segment. During system operation a log or directory is built and written, in updated form, onto a portion of the storage after any text data is written onto the storage. To minimize the time in accessing the pre-determined portion on which the log is stored and to minimize wear of the electromechanical accessing components, portions of more than one segment are dedicated for storage of the log and the segment nearest that on which the text data is written is utilized for storage of the updated log. The beginning of the log includes an activity count indicative of the number of text storage operations that has taken place. The most current log is found by seeking the highest activity count. In addition to providing information relative to segments and portions on which text is stored, the log includes other characteristics to distinguish utilized portions and unused-available portions from portions that are not to be used because of hard errors thereon. Since data is read from the storage and recorded onto the storage by portions, provisions are made for writing into the memory, a number of error codes equal to the number of bytes in a portion when read errors occur in reading from the storage and during relocation for recording a number of error codes equal to a portion when the text data cannot be successfully read from the storage. Thus, the appropriate storage size is maintained for rekeying the text.

## SEGMENTED STORAGE LOGGING AND CONTROLLING

Abstract of the Disclosure

A system and method are disclosed for controlling text storage in a text processing system using a segmented, serial bulk storage device. A plurality of storage segments of equal length are further subdivided into portions of equal length. A typical page of text occupies more than one portion of a segment. During system operation a log or directory is built and written, in updated form, onto a portion of the storage after any text data is written onto the storage. To minimize the time in accessing the predetermined portion on which the log is stored and to minimize wear of the electromechanical accessing components, portions of more than one segment are dedicated for storage of the log and the segment nearest that on which the text data is written is utilized for storage of the updated log. The beginning of the log includes an activity count indicative of the number of text storage operations that has taken place. The most current log is found by seeking the highest activity count. In addition to providing information relative to segments and portions on which text is stored, the log includes other characteristics to distinguish utilized portions and unused-available portions from portions that are not to be used because of hard errors thereon. Since data is read from the storage and recorded onto the storage by portions, provisions are made for writing into the memory, a number of error codes equal to the number of



bytes in a portion when read errors occur in reading from the storage and during relocation for recording a number of error codes equal to a portion when the text data cannot be successfully read from the storage. Thus, the appropriate storage size is maintained for rekeying the text.

#### Cross-Reference to Related Applications

Canadian patent application Serial No. 292,028, filed November 29, 1977, having William C. Cason, et al as inventors and entitled, "Segmented Storage Logging and Controlling For Random Entity Selection".

Canadian patent application Serial No. 292,010, filed November 29, 1977, having Glynn R. Furr as inventor and entitled, "Segmented Storage Logging and Controlling For Partial Entity Selection and Condensing".

#### Background of the Invention

Field of the Invention - This invention relates generally to segmented storage logging and controlling and more particularly to a system and method for the efficient maintenance of segments and segment portions of a segmented serial storage device for the reading and writing of text pages thereon as well as a plurality of stored directories thereon indicating utilization and availability of storage portions.

Description of the Prior Art - Described in U.S. Patents 3,753,239 and 3,781,813 is a technique for logging the utilization of storage blocks on a serial bulk memory and assigning blocks for storage of new and revised text pages to eliminate the burden to a text processing system

1 operator of having to keep track of the logical sequence of  
2 storage blocks that correspond to pages of a document being  
3 prepared or revised. The log was recorded onto a block at  
4 the beginning of the memory at the completion of each storage  
5 or deletion operation.

6 One of the shortcomings of this system is that an  
7 excessive amount of accessing time may be required after a  
8 storage operation to access the storage block on which the  
9 log is to be written or rewritten, which may, for example,  
10 be physically located a relatively large distance away from  
11 the storage block on which the text storage operation took  
12 place. Wear of the electromechanical accessing components  
13 is also increased by the great number of repeated, physically  
14 long distance, accessing moves that are required by the  
15 accessing mechanism in the described system. Nor was the  
16 possibility addressed in these patents that errors might  
17 occur in reading the log from the storage or writing the log  
18 onto the storage. Apparently, if an error occurred in  
19 reading the log from the storage, data on the storage could  
20 be lost such that extensive rekeying by the operator would  
21 be necessary, not because of errors in the recorded text  
22 data on the storage, but because of errors in the log.

23 The above patents disclose a log having a tape log  
24 section thereof in which an indicator bit was provided to  
25 determine the used or unused status of each tape block in  
26 the storage. If a hard error (defect in the tape, itself  
27 that prevents recording or reading) was found in a block,  
28 the block was logged as utilized. Thus, there was no

1 distinction as to whether the block, logged as utilized,  
2 stored valid text data or was not to be used because of  
3 errors.

4 Neither did the above patents address situations that  
5 occur when text data cannot be successfully read from the  
6 storage into the text processing system because of reading  
7 errors. Suppose, for example, that two pages (blocks) from  
8 the tape were read into the text processing memory, but not  
9 reviewed by the operator. Assume that no valid text data  
10 was read from the second block such that this page is not in  
11 the memory. If the operator re-stored this text without  
12 reviewing the text to realize that some of the text was  
13 missing, the log would be updated to reflect that a page of  
14 text was deleted from the job, when, in fact, the operator  
15 needed to maintain storage area to rekey the lost text.

16 It would, therefore, be advantageous to provide a  
17 segmented bulk storage logging and controlling system and  
18 method which overcomes the above shortcomings of the prior  
19 art to provide increased utilization of the storage capability  
20 of the storage device due to improved controlling techniques  
21 and increased integrity of the data and faster access of the  
22 data due to improved logging techniques.

#### 23 Summary of the Invention

24 Accordingly, a system and method are provided for  
25 storing and retrieving text entities for support of a text  
26 processing machine. These text entities, normally representing  
27 pages of a document, are stored as a series of unlabeled,  
28 variable sized members on segments of a serial bulk storage  
29 device. By unlabeled it is meant that no page number or any

1 other logging data is embedded in the text. The system  
2 attempts to maximize the unused storage space by packing the  
3 entities onto the storage segments.

4 The storage is one in which the data is transferred in  
5 a serial manner from or to a segment of storage, but that  
6 segment of storage may be accessed to a "portion" level in  
7 a random or psuedo-random manner. All segments are of fixed  
8 and equal length. Thus, each segment contains a plurality  
9 of portions with each of the portions being of fixed and  
10 equal length.

11 Since the data is stored in a serial, unlabeled format,  
12 the entity (page) number of any stored text cannot be  
13 determined only by inspection of the stored text itself.  
14 Instead, the system maintains a directory which allows it to  
15 ascertain both entity numbers and the locations of these  
16 entities on the storage device. This log (directory) is  
17 comprised of two sections: (1) a system list which is a list  
18 of the logical order of the storage segments utilized and  
19 (2) the system log which is a log of the data characteristics  
20 of every storage segment in the storage.

21 The system attempts to pack entities onto segments of  
22 storage, but not necessarily onto physically contiguous  
23 segments. An entity may cross storage segments and the set  
24 of storage segments in the system list represents a set of  
25 entities (e.g., a multi-page document). The system list is  
26 a logical rather than a physical list of storage segments.  
27 Consider the hypothetical list of storage segments: f/m/d/e.

1 Segment "f" both physically and logically precedes segment  
2 "m". Segment "m" logically, but not physically, precedes  
3 segment "d". The two segments "d" and "e" are physically  
4 contiguous although the two segments "m", "d" and the two  
5 segments "f", "m" are not. Any combination of contiguous,  
6 non-contiguous, physically preceding, or following are  
7 possible in this list. The list implies the logical order  
8 in which the data is stored.

9 In the example, "m" is said to "trail" segment "f" and  
10 segment "d" is said to trail segment "m". Segment "d" does  
11 not trail segment "f". No segment trails segment "e".  
12 Segment "m" is said to "lead" segment "d" and segment "f"  
13 leads segment "m". Segment "f" does not lead segment "d".  
14 No segment leads segment "f".

15 The system log of storage segment data characteristics is  
16 a record of how every portion of every segment is currently used  
17 by the system. Each portion of a segment will have one and  
18 only one of the following data characteristics attributed to it  
19 in the log, as follows: (1) unused-available, (2) error-unusable,  
20 (3) data only, or (4) data and entity begin. A data entity  
21 (page) may span several portions of a storage segment or  
22 several storage segments. No more than one data entity may  
23 exist on a portion. When a data entity spans several  
24 storage segments, they are adjacent in the system list.  
25 When the entity spans portions of a segment, those portions  
26 are serially located on the storage segment. A single  
27 entity is delimited by an entity begin characteristic in the  
28 system log and by a succeeding entity begin characteristic



1 or the end of the system list. Portions on which no text is  
2 stored (error-unusable, unused-available) existing between  
3 entity begins have no effect. The number of a data entity  
4 (page number) is determined by the relative position of the  
5 entity begin in the system list (as determined from the  
6 system log). For example, entity number 3 is located by  
7 finding the third entity begin through inspection of the  
8 system list and the system log of characteristics. As  
9 another example, the retrieval of any entity numbered "i" is  
10 performed by locating the storage segment containing the  
11 "ith" entity begin and reading the correct portion and any  
12 subsequent "data only" portions on that segment or subsequent  
13 segments in the system list until "(1) another ("i + 1")  
14 entity begin portion is located or (2) the list ends.  
15 "Unused" and "error-unusable" portions are skipped.

16 The system directory (log), including both the system  
17 list and the system log is resident in a random access  
18 memory of the system and is recorded on the segmented,  
19 serial storage device. In order to minimize the discrepancies  
20 due to power failure in which the log in random access  
21 memory would be destroyed, the log is recorded on the storage  
22 device each time the storage device has data recorded on it.  
23 To shorten access time during recording, and for reliability,  
24 the log is recorded in multiple locations, physically apart  
25 from each other on the storage device. The physically  
26 closest log is updated at each data storage operation. The  
27 log contains an "activity counter" which is increased at  
28 each update so that the most current, and therefore accurate,  
29 log may be located during system initialization.

1       The system log maintains a record of storage portions  
2       having hard errors thereon so that no further use of these  
3       will be attempted. In addition, a count of these faultly  
4       portions is maintained and the text processing system is  
5       notified when this count reaches a predetermined value.  
6       This notification does not prevent further usage, but serves  
7       as a warning that the storage system's reliability may be in  
8       question.

9       When the text processing system (user) randomly selects  
10      an entity, that entity is subject to revision by the user.  
11      Consequently, the entity may be physically enlarged by text  
12      addition such that it will no longer fit into the space it  
13      previously occupied on the storage device. If the logical  
14      end of this entity shares a storage segment with the next  
15      higher entity number, a possible data overlap problem  
16      exists. Because of this possible conflict, the system will  
17      relocate any data trailing the end of the selected entity  
18      that shares that segment, if the conflicting data exists.  
19      This conflicting data is relocated prior to the initial  
20      reading of the requested entity and is done via the random  
21      access memory buffer which will hold the requested data.  
22      This data is relocated onto another segment which is not  
23      initially included in the system list or system log. The  
24      relocated data will not be logged unless: (1) it is actually  
25      required after the originally requested, revised data has  
26      been stored back onto the storage, or (2) unless errors  
27      occur in attempting to stored the requested data on the  
28      segment from which it originally came, or (3) if entity

1 subsequent to the originally requested entity is also selected.  
2 If the data cannot be relocated, the fetch will still be  
3 performed and the user will be notified of the special  
4 condition. The unlogged segment is called the "scratch  
5 pad".

6 Any data that cannot be read during the relocation is  
7 represented on the "scratch pad" by unique error codes.  
8 Thus, no entity is contracted during the relocation and all  
9 entity begins may be preserved.

10 The foregoing and other objects, features, and advantages  
11 of the invention will be apparent from the following more  
12 particular description of a preferred embodiment of the  
13 invention, as illustrated in the accompanying drawing.

14 Brief Description of the Drawing

15 Fig. 1 is a block diagram of a portion of a text  
16 processing system having a segmented, serial storage device  
17 accessed according to this invention.

18 Figs. 2 and 3 depict the system initialization operation  
19 in which the most recently updated log on the storage is  
20 chosen for use in controlling access to the storage.

21 Figs. 4-9 depict a random entity select operation.

22 Fig. 10 depicts a subsequent entity select operation.

23 Fig. 11 depicts a scratch pad linking operation performed  
24 during both the subsequent entity select and entity record  
25 operations.

26 Fig. 12 depicts a partial read entity select operation.

27 Figs. 13-20 depict the entity record operation.

28 Fig. 21 depicts the update operation used as part of  
29 the entity record operation.

1           Fig. 22 depicts the error update operation used in the  
2   entity record operation.

3           Description of the Preferred Embodiment

4           Referring now to Fig. 1, a portion of a text processing  
5   system is shown having a processor 1 to which is connected  
6   an address buss 2 on which instructions in a control storage  
7   3 are addressed to be provided back to the processor along  
8   an instruction buss 4. A system clock generator 5 provides  
9   clock signals along lines depicted by C to each of the  
10   devices in the system. The processor 1 transmits control  
11   signals along line 6 to a segmented serial storage device 9  
12   which may, for example, comprise a disc on which the various  
13   tracks thereof are segments, or storage 9 may comprise, for  
14   example, a tape on which blocks thereof are segments.  
15   Storage 9 provides interrupt and status information along  
16   lines 7 and 8, respectively, back to the processor 1.

17          A random access memory 21 is provided to store text  
18   data for text creation and revision purposes and to serve  
19   as a buffer in relocating data on storage 9. Random access  
20   memory controller 18 controls access to the memory via  
21   enable line 17, data buss 19 and address buss 20. Transfers  
22   of data directly between the storage 9 and memory 21, without  
23   invoking the processor 1, are accomplished by use of the  
24   direct memory access controller 13. Data is transferred  
25   between the direct memory access controller 13 and the  
26   random access memory controller 18 via data buss 15 and is  
27   transferred between storage 9 and the direct memory controller  
28   13 via data buss 12. The address buss 14 points to the

1 appropriate address in the random access memory 21 to which  
2 data is to be written or from which data is to be read in  
3 accordance with an enable signal on line 16. Control signals  
4 from the direct memory access controller to the storage 9  
5 and status signals from storage 9 to the direct memory  
6 access controller 13 are conveyed along lines 11 and 10,  
7 respectively. The processor 1 conveys the appropriate  
8 address for reading or writing to the random access memory  
9 controller 18 via address buss 22. The data is transferred  
10 between the processor and the random access memory controller  
11 via data buss 23 in accordance with an enable signal on line  
12 24.

13 Control storage 3 is typically implemented in read only  
14 storage with the instructions, therefore, permanently wired  
15 thereinto. However, control storage may be implemented in  
16 the form of a random access memory such that the instructions  
17 must be loaded thereinto each time power is applied to the  
18 system. In another embodiment, the processor 1 and control  
19 storage 3 may be replaced entirely by combinational logic  
20 such that no processor or "instructions" as such are utilized.  
21 The flow diagrams described hereinafter will enable any  
22 person having skill in the art of logic design to specify  
23 hardware logic in accordance with the concepts of this  
24 invention. These flow diagrams will also enable anyone  
25 having skill in the art of computer programming to program a  
26 general purpose digital computer to access a segmented  
27 serial storage device and log the utilization of this device  
28 in accordance with the concepts of this invention.

1           Referring now to Fig. 2, the system initialization  
2 process is shown to examine a plurality of logs stored on  
3 storage 9 to find the most current log in accordance with  
4 the count in an activity counter field thereof. An initial  
5 assumption is that there are "X" number of directories or  
6 logs on storage device 9. Block 31 indicates that registers  
7 COUNT and READOK are set to zero and register N is set to  
8 one. In block 32 the system directory number "N" is read  
9 from storage 9. At 33 the number read is tested for read  
10 errors. If the read was unsuccessful, register N is incremented  
11 upwardly by one at 36, and at 37 the contents of the N  
12 register are compared with "X". Assuming that the N register  
13 contents do not exceed "X", there are other directories  
14 to be read and the operation is repeated by reading the  
15 next system directory number "N" from storage 9, as shown at  
16 32. Assuming, at 33, that the read was successful, the  
17 READOK register is set to one at 34 and the count of the  
18 activity counter stored at the beginning of the directory is  
19 compared against the contents of the COUNT register at 35.  
20           If the activity counter contents are greater than or  
21 equal to zero, the directory is installed into the random  
22 access memory 21 as indicated by block 38. (In this example,  
23 with the first compare of the activity count or contents  
24 with the COUNT register contents, the activity counter will  
25 always be equal to or greater than the COUNT register contents,  
26 since the COUNT register was initially set to zero.) At 39,  
27 the count register is loaded with the activity counter  
28 contents and at 36 the register N contents are incremented  
29 by one. A directory is installed into the memory 21 each

1 time the contents of its activity counter exceed the contents  
2 of the COUNT register. At 37, when all of the directories  
3 have been read, the directory in the random access memory 21  
4 will be the one that was successfully read having the highest  
5 activity counter contents.

6 Continuing from Fig. 2 to Fig. 3, when all of the  
7 directories have been read, the READOK register is tested at  
8 40 to determine whether its contents are 0 or 1. If the  
9 contents are 1 a good directory was read from storage 9 and  
10 the operation returns to the user text processing system at  
11 block 42. If a good directory was not read, the user is  
12 notified of the absence of a valid directory at 41, which  
13 will prevent further access of the storage 9.

14 Refer next to Figs. 4-9 which show a random entity  
15 select operation in which a random page, specified by the  
16 operator, and, in turn, by the text processing system to the  
17 logging and controlling system, is fetched from the storage  
18 9 and loaded into the random access memory 21. The operation  
19 begins in Fig. 4 with the following assumptions. For this  
20 example the requested entity number is contained in the N  
21 register and is denoted N, the random access memory buffer  
22 initial address to which data from storage 9 will be loaded  
23 is denoted B and stored in a B register, and the unused  
24 capacity of memory 21 is denoted as M and stored in an M  
25 register. The capacity M is an integer indicating the size  
26 of memory 21, buffer B. This is the number of whole portions  
27 of storage 9 that buffer B can contain under given cir-  
28 cumstances. For a random entity select, M must be at least

1 the number of portions on a segment minus 1. For this  
2 example, assume that each segment of storage 9 includes 8  
3 portions. Thus, M must be at least 7.

4 When the operation is begun, a test is made at 46 to  
5 see if entity N exists by scanning the system log portion of  
6 the directory and counting the data and entity begin portions  
7 thereof. If the data entity N does not exist, obviously no  
8 text data can be read from storage 9 into the random access  
9 memory 21. However, the following operation is performed to  
10 set the storage controlling system to store any additional  
11 text following the last previously stored entity.

12 A concept incorporated in the storage controlling  
13 system is best defined at this point --the select pointer/record  
14 pointer concept. The system maintains two register-stored  
15 "pointers" to locations on the segmented serial storage  
16 device 9. These pointers each include a field corresponding  
17 to a segment on storage 9 and another field corresponding to  
18 a particular portion of that segment. The select pointer,  
19 denoted SP, is defined to indicate (e.g., point to) the next  
20 portion of a segment to be accessed or fetched. Since the  
21 system has the capability to partially read an entity (as  
22 will be shown in detail in subsequent flow diagrams) and to  
23 complete this partial read at a later time, the SP does not  
24 always point to an entity begin. When both the segment  
25 field and the portion field of the SP are 0, all data has  
26 been selected. Entities may only be created, not revised,  
27 when the entire SP is 0.



1       The record pointer, denoted RP, points to the last  
2       portion of a segment that was recorded onto the storage  
3       device. When a record pointer has the portion field of it  
4       equal to 0, the segment it indicates in the segment field  
5       has not been recorded upon on any portion. The RP cannot  
6       ever, "logically" be greater than or equal to the SP as  
7       determined by the system list and system log. (For this  
8       definition when the select pointer register contents equal  
9       zero, the select pointer is considered to be nonexistent).  
10      The pointers may indicate the same storage segment, but not  
11      the same portion of that segment.

12      At the conclusion of each entity store operation, the  
13      area "logically" between the RP and SP has the data characteristics  
14      thereof set to unused in the system log (with the exception  
15      of error-unusable portions). When the select pointer is  
16      zero, the area between the RP and the end of the system list  
17      is set to unused in the system log. Every segment which has  
18      no data on it is removed from the system list. This insures  
19      that additional storage space will be available for subsequent  
20      work.

21      Continuing the above case, Fig. 4, in which the entity  
22      does not exist, at 49 both fields of the select pointer SP  
23      register are set to zero. A SCRATCH PAD register is set to  
24      empty. At 50 the segment field of the record pointer RP  
25      register is set to the last logical segment in the system  
26      list with data on it and, at 51, the portion field of the RP  
27      register is set to the last portion having text data on it.

1 of the segment number denoted by the segment field portion  
2 of the RP. At 52, the user is notified that the entity was  
3 not found. Thus, the select pointer has been set to indicate  
4 that there is no further stored text to be selected and a  
5 record pointer has been set such that text in the random  
6 access memory 21 may be stored immediately succeeding the  
7 last stored entity in storage device 9.

8         Returning to 46, consider the more probable case in  
9 which the specified entity N does exist. If the next entity,  
10 N+1, begins on the same segment of storage on which entity N  
11 ends, a possible conflict can occur. The problem is that if  
12 the text of entity N is accessed and enlarged, it will no  
13 longer fit on the same storage area from which it came. If  
14 the next entity does not begin on the same segment on which  
15 the specified entity N ends, there can be no conflict, since  
16 any expansion of the text beyond the segment boundary could  
17 be stored on a previously unused segment. Referring now to  
18 block 47, Fig. 4, consider the case in which no conflict can  
19 exist. At 48, the SCRATCH PAD register is set to empty and  
20 the operation proceeds to C Fig. 7. At 70, the segment  
21 field of the SP register is set to the number of the first  
22 segment on which entity N is stored. This information is  
23 obtained from the directory. At 71, the portion field of  
24 the SP register is set to the number of entity N's first  
25 data portion from information obtained from the system log  
26 portion of the directory. At 72, if the entity to be accessed  
27 is the first entity of a document, at 73 the segment field

1 of the RP register is set to the segment number of storage 9  
2 containing the first segment of entity N at 73 and, at 74  
3 the portion field of the RP register is set to zero.

4 Before proceeding to Fig. 8, assume that the requested  
5 entity N was not the first entity, such that the operation  
6 continues at 75 wherein the segment field of the RP register  
7 is set to the number of the last data segment of entity N-1  
8 and, at 76 the portion field of the RP register is set to  
9 the number of the last portion on which data of entity N-1  
10 was stored.

11 Proceeding to D, Fig. 8, the following decision at 78  
12 determines if a partial read situation occurs by determining  
13 whether there is sufficient room in memory 21 to contain  
14 whatever section of the entity that may exist past the  
15 select pointer but on the same segment as indicated by the  
16 segment field of the select pointer. At 78, if M (memory  
17 size by portions) is large enough to contain whatever portions  
18 of entity N exists on the segment specified by the segment  
19 field of the SP register, at 79 text portions of the SP  
20 segment containing entity N are read into the random access  
21 memory 21 at address B. At 80, if there are no read errors,  
22 the operation proceeds to D1, as subscribed hereinafter. If  
23 there are read errors, at 81 error codes are input to  
24 buffer B for each storage location in each portion having  
25 read errors to provide an indication to the operator as to  
26 where the read errors occurred, and the operation proceeds  
27 to D1.

1           At 78, Fig. 8, assuming that M was not large enough to  
2 contain entity N as it exists on the SP segment, at 82 the  
3 user is notified of the heretofore described partial read  
4 situation. At 83, if M is zero, indicating that the memory  
5 is full, the operation proceeds to D1 as described hereinafter,  
6 and, if M is not equal to zero, at 84, the operation proceeds  
7 by reading as much of entity N on the SP segment as memory  
8 21 can hold. Any data read into memory 21 is tested for  
9 errors with error codes being input if necessary, in accordance  
10 with 80 and 81, previously described.

11           Referring now to D1, Fig. 9, at 86 assume that a  
12 partial read was necessary because of insufficient buffer  
13 capacity. At 87, the select pointer register is set to the  
14 next unread segment and portion of entity N so that reading  
15 can continue from this point when space is available in  
16 memory 21. At 88, if there are any error codes in the data  
17 that was written into the memory 21, the user is notified at  
18 89. If not, the system returns to the user.

19           In Fig. 9, at 86 consider the case in which there was  
20 not a partial read of data. At 91, the M register is decremented  
21 by the number of portions of data just read. At 92, if the  
22 entity N+1 exists on the SP segment, all of entity N has  
23 been read, and at 93 the portion field of the SP register is  
24 set to the entity begin portion of entity N+1. Thus, the  
25 select pointer now points to the beginning of the next  
26 entity in the logical sequence of entities in the document.  
27 At this time the test is made for error codes in memory 21  
28 and the user is notified if there are error codes.

1           At 92, if entity N+1 does not exist on the SP segment,  
2     the system list portion of the directory is tested at 94 to  
3     see if a segment trails the segment specified by the segment  
4     field of the SP register. If not, at 98 the SP register is  
5     set to zero which indicates that there are no further entities  
6     that can be read. The system is then returned to the user  
7     after notification of any error codes previously read into  
8     memory 21, if any. At 94, if a segment does trail the  
9     select point segment, the segment field of the SP register  
10    is set to the trailing segment and the portion field of the  
11    SP register is set to the first data portion on that trailing  
12    segment at 95, this information being obtained from the  
13    directory. At 96, if the portion field of the SP register  
14    is now set on the portion of an entity begin, then reading  
15    of the requested entity has been completed, and the system  
16    is returned to the user after notification of any error  
17    codes in memory 21, if any. If the portion field of the SP  
18    register is not pointing to an entity begin on the trailing  
19    segment, at 97 address B is incremented by the amount of  
20    data just read and the operation proceeds to D which will  
21    continue reading on the next segment.

22           Referring back to Fig. 4 at 47, assume that there is a  
23    possible conflict. That is, the next entity (N+1) shares a  
24    segment with the requested entity. Again, the conflict will  
25    occur if the requested entity is expanded in size such that  
26    there is not sufficient room on the segment in which the N  
27    entity ends to complete recording of the expanded N entity  
28    without writing the data in the portions storing the N+1 entity.

1 Proceeding now to B, Fig. 5, at 55 an empty segment is located  
2 on storage 9 and defined as ES. The empty segment is found  
3 by referring to the directory to identify a segment which  
4 has neither data nor error-unusable portions in the system  
5 log and is not in the system list portion of the directory.  
6 At 56, assume now that there was an available empty segment  
7 on storage 9 so that at 57 the conflicting data from the  
8 last segment containing entity N is read into address B of  
9 memory 21. This conflicting data is all data on the segment  
10 on which entity N ends that succeeds the end of entity N.  
11 Memory 21 serves as a temporary buffer for this data to  
12 allow this data to be copied onto the ES segment. At 58 if  
13 any errors are encountered in reading the conflicting data,  
14 error codes are written into memory 21 for all portions read  
15 from storage 9 that have read errors, as indicated at 59.  
16 After the error codes have been written, or if there were no  
17 error codes, at 60 the conflicting data is read back out of  
18 memory 21 and is written onto the ES segment, as indicated  
19 at 60.

20 The writing operation is checked for write errors at  
21 61, and if there are any write errors the data characteristics  
22 of the ES segment just written on are temporarily set to  
23 indicate data in each portion thereof, at 62, to prevent  
24 further reuse of this segment in attempting to relocate the  
25 data. The operation then proceeds back to 55 at which time  
26 another empty segment is attempted to be located on which to  
27 relocate the conflicting data.

28 At 61, assuming that no write errors occurred in  
29 relocating the data, the operation proceeds to 31, Fig. 6.

1 At 65, the SCRATCH PAD register is set to the segment ES.  
2 At 66, the necessary information about segment ES is saved  
3 so that it can be installed in the directory in the future  
4 if it is ultimately needed. (It will be needed, for example,  
5 if the recalled data is expanded such that it would, otherwise,  
6 overwrite the conflicting data now stored on the scratch pad  
7 segment.) The information about the ES segment that is  
8 saved is the segment number from which the relocated data  
9 was copied is stored in an A register and defined as A, and  
10 the data characteristics (unused-available, error-unused,  
11 data, or data and entity begin) of the ES segment as it now  
12 exists. Proceeding now to 67, any segments that encountered  
13 write errors during the relocation of the data are now  
14 released. In Fig. 5, block 62, it will be recalled that  
15 segments on which write errors occurred were set in the  
16 system log to indicate data on all portions thereof. Releasing  
17 the segment, therefore, implies resetting these data portions  
18 in the system log to unused. The segments were never put in  
19 the system list.

20 After any segments in which writing errors were encountered  
21 are released, the operation continues at C, Fig. 7 to allow  
22 the requested entity to be written into memory 21. Referring  
23 back to Fig. 5, if a possible conflict existed to invoke an  
24 attempt to relocate the conflicting data, if no empty  
25 segment is located at 56, the scratch pad is set to empty and  
26 the user is notified of the inability-to-relocate status in  
27 the system at 63. Operation then proceeds to B2, Fig. 6.  
28 After any segments are released that encountered write  
29 errors the operation proceeds to C as described above.

1           Another category of system operation is the subsequent  
2       entity select which assumes that a random entity select  
3       operation has previously occurred. The point here is that  
4       the select pointer has been previously set at the end of the  
5       previous entity select operation to point to the entity that  
6       will be accessed by the subsequent entity select operation,  
7       there being no initialization of the select pointer at the  
8       beginning of the subsequent entity select operation. In  
9       Fig. 10 the operation begins by a request from the user that  
10      entity N be read into buffer address B of memory 21 which  
11      has a capacity M. At 101, the system log portion of the  
12      directory is interrogated to determine if entity N does in  
13      fact exist. If not, the select pointer register is set to  
14      zero at 103, and at 104 the user is notified that the entity  
15      was not found in storage 9. At 105, the storage controlling  
16      system is then returned to the user.

17           Assume now that at 101 the entity was found to exist in  
18      the system log portion of the directory. The subroutine  
19      SCPAD (Scratch Pad) is called into the operation at 102.

20           Referring now to Fig. 11, the SCPAD operation is  
21      described. At 108, if the SCRATCH PAD register is empty,  
22      the operation returns to the caller, which point is at D,  
23      Fig. 10. Thus, the operation then proceeds to D at Fig. 8  
24      to proceed with the reading of data in the subsequently  
25      selected entity into the memory 21. If, however, the SCRATCH  
26      PAD register is not empty, the scratch pad segment is logically  
27      linked into the list of segments in the system list, as  
28      follows. At 109 the system list portion of the directory is



1 updated to insert the scratch pad segment number trailing  
2 the number of segment A. Segment A is the segment from  
3 which data trailing a previously selected entity was copied  
4 to the scratch pad. At 110 the system log portion of the  
5 directory is updated to reflect the actual data characteristics  
6 of the scratch pad segment. At 111, the system log portion  
7 of the directory is updated to reflect the new data characteristics  
8 of segment A. Portions of segment A copied onto the scratch  
9 pad segment will now be listed as unused in the system log.  
10 At 112, the select pointer SP register segment field is set  
11 to the scratch pad segment. At 113, the SCRATCH PAD register  
12 is set to empty and the operation returns to the caller,  
13 point D, Fig. 10. From this point, reading of the requested  
14 entity into the memory 21 continues. It will be noted that  
15 reading is from the data on the scratch pad segment and not  
16 from the data on segment A from which the scratch pad data  
17 was copied.

18 Another system operation is the partial read entity  
19 select, the start of which is shown as Fig. 12. In this  
20 case, a request from the user is that reading of a previously  
21 partially read entity be continued from storage 9 into  
22 memory 21 at buffer address B. When the buffer capacity M  
23 becomes larger than 0, reading of the entity, or a portion  
24 thereof, can continue at D, Fig. 8.

25 A major operation of the storage logging and controlling  
26 system takes place when an entity must be recorded onto  
27 storage 9. Appropriate segments and portions thereof must  
28 be chosen for the recording and this information must be  
29 logged. Referring now to Fig. 13, a recording operation

1 begins by a request from the user that the contents of  
2 memory 21 at address B be stored on the storage 9 at the  
3 entity N. The size of the entity is M portions.

4 At 116, the system log portion of the directory is  
5 interrogated to determine if any entities exist on storage  
6 9. If no entities exist, an empty segment must be chosen  
7 for recording and the select pointer, scratch pad, and  
8 record pointer registers must be set. At 117, an empty  
9 segment is found and placed in the system list with area  
10 of the directory. At 118, the SP register is set to 0 and  
11 the SCRATCH PAD register is set to empty. Then at 119 the  
12 segment portion of the RP register is set to the segment in  
13 the system list and the portion field of the RP register is  
14 set to zero. The operation continues at 120. At 116,  
15 if an entity does exist the operation proceeds directly to  
16 120. If an entity does exist, the assumption is that the  
17 record and select pointers have already been set since it is  
18 assumed that when entities do exist, an attempt to select  
19 one of them must have been made before a recording process  
20 can take place.

21 Now at 120 the SP register is tested for zero status.  
22 The SP register will be zero if no entities previously  
23 existed or if there are existing entities with the record  
24 pointer pointing to the last portion in the existing entities.  
25 In other situations, the select pointer register will not be  
26 zero and operation proceeds to H, Fig. 14.

27 With the select pointer not equal to zero, the select  
28 pointer is pointing to a next entity of one or more entities

1 and the record pointer precedes the select pointer. Recording  
2 will start at the record pointer, but must stop before  
3 actually reaching the select pointer. Thus, in Fig. 14, at  
4 122, the SCRATCH PAD register is tested to see if it empty.  
5 If it is empty, there is not even a potential conflict in  
6 which the record pointer might reach the select pointer so  
7 the operation proceeds to G to continue the recording operation  
8 as will be described hereinafter.

9 If the scratch pad is not empty, a determination must  
10 be made as to whether the text that originally preceded the  
11 scratch pad contents has been expanded so that it will no  
12 longer fit back onto the segment containing an entity that  
13 begins after the previously selected entity that is to be  
14 stored. Thus, assuming that the scratch pad is not empty,  
15 at 123 a test is made to determine if there is enough space  
16 between the record pointer and the select pointer to contain  
17 entity N. If there is enough space, the recording operation  
18 can proceed to G. If there is not enough space, the subroutine  
19 SCPAD as previously described in Fig. 11, is called into the  
20 operation. Calling this subroutine places the scratch pad  
21 into the system list and system log so that the current  
22 recording process can now write into the now unused portion  
23 of the record pointer and select pointer segment.

24 The operation now proceeds to G, Fig. 15, or would have  
25 proceeded directly to G, if, at Fig. 13, 120, the select  
26 pointer had been at zero. At 126 the portion field of the  
27 RP register is tested to determine if the record pointer is  
28 pointing to the end of a segment. If so, the operation

1 proceeds to I, described hereinafter, at which time another  
2 segment must be found for recording, if any exists. If the  
3 record pointer is not pointing to the end of a segment, at  
4 127 the contents of memory 21 are tested to see if they are  
5 zero. This is part of the recording operation, since the  
6 operation of restoring data portions and segments from used  
7 to unused status is to access this data from storage 9 into  
8 memory 21, delete this data from memory 21, and request  
9 recording after the data has been deleted.

10 Assume that it is desired to delete data from storage  
11 so that the contents of memory 21 are zero, at 127, Fig. 15,  
12 the operation proceeds to J, Fig. 16. At 133, Fig. 16, all  
13 portion fields of the system log area of the directory  
14 between, but not including the record pointer position and  
15 the select pointer position are set to unused. The status  
16 of any logged error portions between the record pointer and  
17 the select pointer is not changed. At 134, the directory is  
18 searched to determine if there are any empty segments in the  
19 system list. If so, at 135 these segments are removed from  
20 the system list so that the system will consider them to be  
21 usable empty segments. The operation then proceeds to L,  
22 Fig. 18. If there are no empty segments in the system list,  
23 the operation proceeds directly to L.

24 In Fig. 18, all that remains in this deletion portion  
25 of the record operation is to update the directory and  
26 record the updated directory on storage 9. At 148, a counter  
27 Y is initialized to one to count the directories recorded on  
28 storage 9. At 149, the activity counter at the very beginning

1 of the directory in the random access memory 21 is incremented  
2 by 1. At 150, the directory is then recorded at the dedicated  
3 directory recording area on storage 9 that is physically  
4 closest to the current position of the read/write transducer  
5 associated with storage 9. At 151, if no record error  
6 occurs during recording of the directory, the operation is  
7 returned to the user. If there is a record error at 151,  
8 counter Y is upwardly incremented one count at 152 and a  
9 test is made at 153 to determine whether the counter Y  
10 contents exceed the number of directories X. If Y does not  
11 exceed X, there are other directory recording areas on  
12 storage 9 and another attempt is made at 154 to record the  
13 directory in the next closest dedicated directory recording  
14 area. If Y exceeds X there are no other directory recording  
15 areas and the user is notified that the directories are in  
16 error and that the storage 9 media is no longer usable.

17 Referring back to Fig. 15, the immediately preceding  
18 discussion has been directed to the deletions portion of the  
19 record operation wherein M equals zero at 127. Assume now  
20 that text from memory 21 is to be stored back onto storage 9  
21 such that M is not equal to zero. The record pointer is now  
22 positioned for recording and, at 128, starting at the first  
23 portion following the record pointer, as much data from  
24 memory 21 is recorded as possible on the non-error portions  
25 of the segment pointed to by the segment field of the record  
26 pointer. Recording is stopped just before the portion  
27 indicated by the portion field of the select pointer if the  
28 select pointer is encountered on this segment. At 129 the

1 newly recorded data on storage 9 is tested for record errors.  
2 If there are no record errors, the UPDATE subroutine operation  
3 is called at 130.

4 Referring now to Fig. 21, the UPDATE operation is shown  
5 wherein at 176 M (the number of portions of data in memory  
6 21) is decremented by the amount of data just recorded. At  
7 177 memory address B is incremented by the amount of data  
8 just recorded. The system log is updated to reflect any new  
9 data or data and entity begin portions that were used during  
10 the recording operation. Finally, at 179 the portion field  
11 of the RP register is set to the last portion on which  
12 recording was attempted, regardless of whether that portion  
13 was an error portion or not. At this time, the operation is  
14 returned back to 131, Fig. 15.

15 Assume that at 129, Fig. 15, there were record errors  
16 in the data just recorded. The operation proceeds to K, Fig.  
17 17. First, at 138 the SCPAD (scratch pad) subroutine is  
18 called which is shown in Fig. 11 and has been previously  
19 described. If the scratch pad is empty, the operation  
20 returns immediately to 139 and if the scratch pad is not  
21 empty, the scratch pad segment is linked into the directory  
22 and the operation proceeds at 139. At 139, if any additional  
23 empty segments exist on storage 9, the ERRUP (error update)  
24 subroutine is called at 145.

25 Referring now to Fig. 22, the ERRUP subroutine begins  
26 at 182 by updating the system log portion of the system

1 directory to indicate the error portion. At 183, the system  
2 error portion counter is incremented for each error portion  
3 logged. At 184, a test is made to determine whether the  
4 contents of the error counter are equal to the error limit.  
5 The error limit is a predetermined number one unit greater  
6 than the number of error portions that will be allowed  
7 without cautioning the user. If this limit has been reached,  
8 at 185 the user is notified of the error limit. After  
9 notification, or if the error count has not reached the  
10 limit, the operation returns to the caller at Fig. 17, point  
11 G. Then, at Fig. 15, subsequent recording attempts are  
12 made, if indicated.

13 In Fig. 17, assume at 139 that no further empty segments  
14 exist. At 140, a test is made to see if an entity begin for  
15 the entity N being recorded has been logged. If the entity  
16 begin has been logged, ERRUP subroutine in Fig. 22 will  
17 be called. At the completion of this subroutine no further  
18 data recording is attempted and the user is notified that  
19 all data to be recorded has not been recorded at 144. Then  
20 the operation proceeds to L, Fig. 18 as previously described,  
21 so that the directory can be updated and re-recorded onto  
22 storage 9. At 140, if an entity begin representing entity N  
23 has not been logged, the portion field of the record pointer  
24 is set to the next sequential portion on this segment at  
25 142 and, at 143 the entity begin is forced. This prevents  
26 the loss of entity begins and implicit renumbering of sub-  
27 sequent entities.

28 Referring back to Fig. 15, the operation from point I  
29 will now be discussed in which case another segment must be  
30 accessed for recording. Referring now to Fig. 19, at 158 the  
31 system list portion of the directory is scanned to determine

1 whether a segment in the system list trails the segment  
2 pointed to by the segment field of the RP register. If a  
3 segment does trail, at 159 a determination is made as to  
4 whether the trailing segment is eligible for recording. The  
5 trailing segment will be considered to be eligible for  
6 recording if: (1) the segment portion of the SP register  
7 contents do not point to the trailing segment and (2) the  
8 trailing segment contains no entity begins. If the trailing  
9 segment is eligible, at 160, the segment portion of the RP  
10 register is set to the trailing segment number and the  
11 portion field of the RP register is set to zero. This will  
12 enable recording to continue from the beginning of this  
13 trailing segment in accordance with the operation starting  
14 at G, Fig. 15.

15 If the trailing segment is not eligible for recording,  
16 at 161 the SCRATCH PAD register contents are tested. If the  
17 scratch pad is not empty, an exception to not recording on  
18 an ineligible trailing segment is invoked at 160. At this  
19 point, the segment field of the RP register is set to the  
20 trailing segment and the portion field of the RP register is  
21 set to zero so that recording can proceed to G. This segment  
22 will be recorded upon since the scratch pad can be later  
23 linked if it is needed due to errors in recording in this  
24 trailing segment.

25 At 161, if the scratch pad is empty, the directory is  
26 interrogated at 162 to determine if an empty segment exists.  
27 If an empty segment does exist, the empty segment is linked



1 into the system list trailing the current RP segment and the  
 2 operation continues at 160, as described above. If an empty  
 3 segment does not exist, the operation proceeds to I2, Fig.  
 4 20. Before leaving Fig. 19, it will also be noted that at  
 5 158, if a segment does not trail the RP segment in the  
 6 system list, the operation jumps to 162 to determine whether  
 7 or not an empty segment exists. This is the case in which  
 8 the record pointer was pointing to the last segment in the  
 9 system list.

10 Referring now to I2, Fig. 20, at 166 the storage space  
 11 between the record pointer and the select pointer, not  
 12 inclusive, is calculated. If the select pointer is set to  
 13 zero, then the amount of space equals zero by definition.  
 14 At 167, if the amount of space equals zero, the operation  
 15 proceeds to K1, Fig. 17. In this situation the user is  
 16 notified that all data is not recorded and the operation  
 17 proceeds to update the directory and re-record a directory  
 18 onto storage 9. If the amount of space in storage 9 between  
 19 the record pointer and the select pointer is greater than M,  
 20 (the memory 21 contents) then the operation proceeds to I1,  
 21 Fig. 19. From there the operation proceeds to 160, Fig. 19,  
 22 as described above. In Fig. 20, at block 168, if the available  
 23 space on storage 9 between the record pointer and the select  
 24 pointer is less than the memory 21 contents, the user is  
 25 notified that all data was not recorded at 169 and the  
 26 operation proceeds to 170 where M is set equal to the calculated  
 27 storage space. The operation then proceeds to I1, Fig. 19.

1 In Fig. 19, the record pointer segment field is set to the  
2 trailing segment and the record pointer portion field is set  
3 to zero. The operation then proceeds to G, Fig. 15 for  
4 recording to continue from the record pointer toward the  
5 select pointer. The system halts recording before the  
6 record pointer overruns the select pointer.

7 In summary, therefore, a system and method are provided  
8 for storing and retrieving text entities and portions thereof  
9 for support of a text processing machine. These text  
10 entities, normally representing pages of a document, are  
11 stored as a series of unlabeled variable sized members on  
12 segments of a serial bulk storage device. The system attempts  
13 to maximize the unused system storage space by packing the  
14 entities onto the storage segment. The system directory,  
15 including both the system list and the system log, is resident  
16 in a random access memory of the system and is recorded on  
17 the segmented, serial storage device. The log is recorded  
18 on the storage device each time the storage device has data  
19 recorded onto it. To shorten access time during recording,  
20 and for reliability, the log is recorded in multiple locations,  
21 physically apart from each other on the storage device. The  
22 physically closest log is updated at each data storage  
23 operation. The log contains an "activity counter" which is  
24 increased at each update so that the most current, and  
25 therefore accurate log may be relocated during system  
26 initialization.

27 The system log maintains a record of storage portions  
28 having hard errors thereon so that no further use of these

1 portions will be attempted. Additionally, a count of these  
2 faultly portions is maintained and the text processing  
3 system is notified when this count reaches a predetermined  
4 value.

5 In accessing a selected entity of data, the system  
6 relocates any data trailing the end of the selected entity  
7 that shares the segment on which the selected entity ends.  
8 This conflicting data is relocated prior to the initial  
9 reading of the requested entity and the relocation is done  
10 via the random access memory buffer which will ultimately  
11 hold the requested data. The data is relocated onto another  
12 storage segment which is not initially included in the  
13 system list. The relocated data will not be logged unless:  
14 (1) it is actually required after the originally requested,  
15 revised data has been stored back onto the storage, (2) or  
16 unless errors occur in attempting to store the requested  
17 data onto the segment from which it originally came, or (3)  
18 if the entity subsequent to the originally requested entity  
19 is also selected. If the data cannot be relocated, accessing  
20 of the requested entity will still be performed and the  
21 system will be notified of this special condition. The  
22 unlogged segment is called the scratch pad. Any data that  
23 cannot be read during the relocation is represented on the  
24 scratch pad by unique error codes. Thus, all entity begins  
25 are preserved during the relocation.

26 When the memory will not hold the entire requested  
27 entity, the partial entity select operation is used to  
28 transfer as many portions of an entity as there are

1 corresponding whole integer portions in the text processing  
2 system memory, from the storage device to the memory, and  
3 to continue fetching the remainder of an entity to the text  
4 processing system memory when the memory has room to accept  
5 additional text.

6 Condensing of the segments is accomplished by alternate  
7 reading of an entity followed by recording of the entity  
8 with no other revision between reading and recording. This  
9 moves unused portions to build whole unused segments so that  
10 these unused segments may be deleted from the system list.

11 While the invention has been particularly shown and  
12 described with reference to a preferred embodiment thereof,  
13 it will be understood by those skilled in the art that  
14 various changes in form and details may be made therein  
15 without departing from the spirit and scope of the invention.

1       The embodiments of the invention in which an exclusive property or privilege is claimed are defined as follows:

1.   A method of storing machine logging data indicative of the content of all segments and portions of said segments of a storage media for text storage in a text processing system, comprising:

dedicating a plurality of defined portions of said segments for storage of said logging data on said dedicated portions;

10   storing initial text data on said media and the logging data indicative of the then content of said media on only one of said dedicated portions; and

storing, at the termination of each storage of updated text data on said media, the most current logging data on only one of said dedicated portions.

2.   The method of claim 1 wherein said step of storing said logging data on only one of said dedicated portions further comprises storing said logging data on the one of said dedicated portions physically closest to the storage  
20 reading and recording transducer used for said storage of text data on said media.

3.   The method of claim 2 wherein said step of dedicating a plurality of segments of said media further comprises separating said dedicated portions by one or more of said storage media segments dedicated for text storage.

4.   The method of claim 3 wherein said step of storing said logging data further comprises storing an activity counter number included with said logging data which number  
30 is made one unit larger each time text data is stored on said media.

- 1           5.    The method of claim 4 wherein said step of storing  
said logging data further comprises storing a first data  
characteristic indicative of the unused-available status  
of segment portions and a second data characteristic indica-  
5    tive of the error-unusable status of error portions.



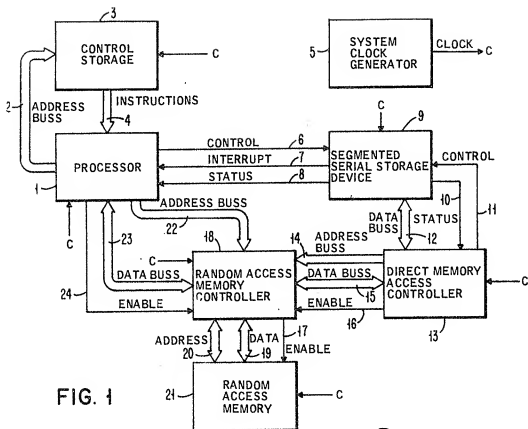


FIG. 1

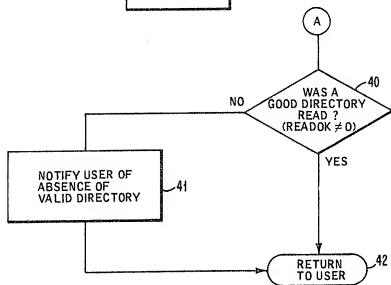


FIG. 3

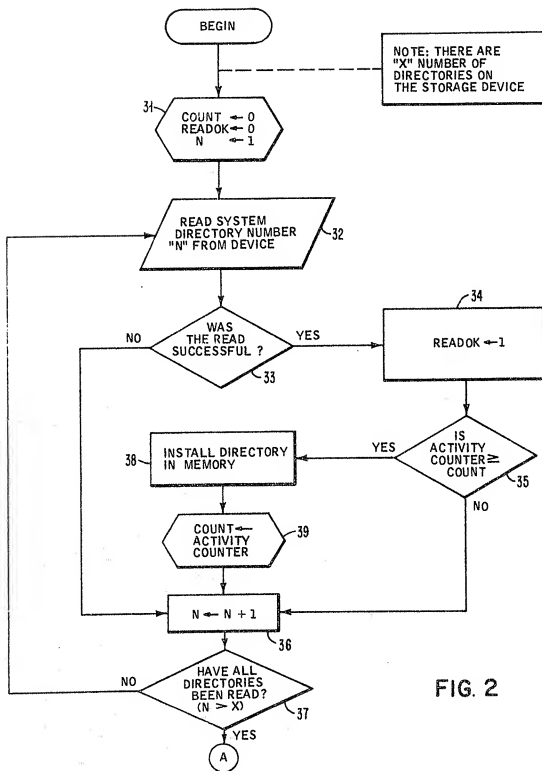
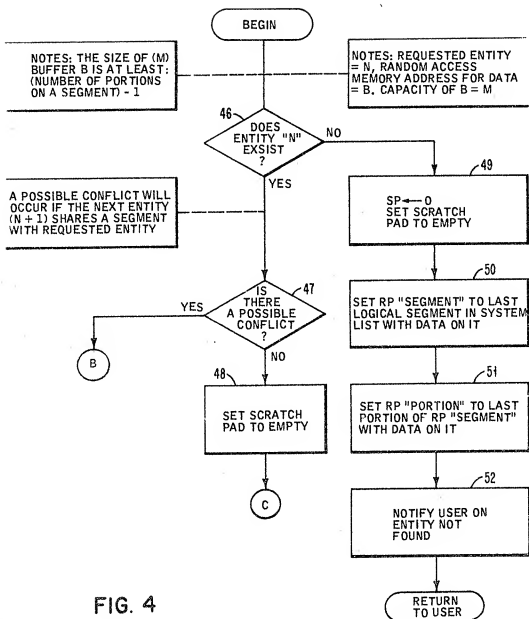


FIG. 2





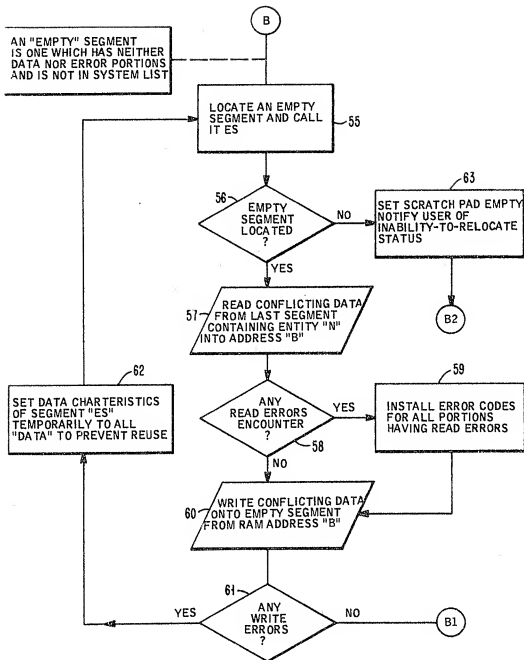


FIG. 5

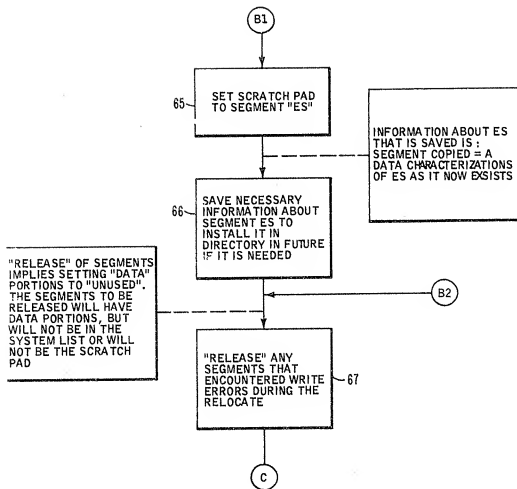


FIG. 6

19-6

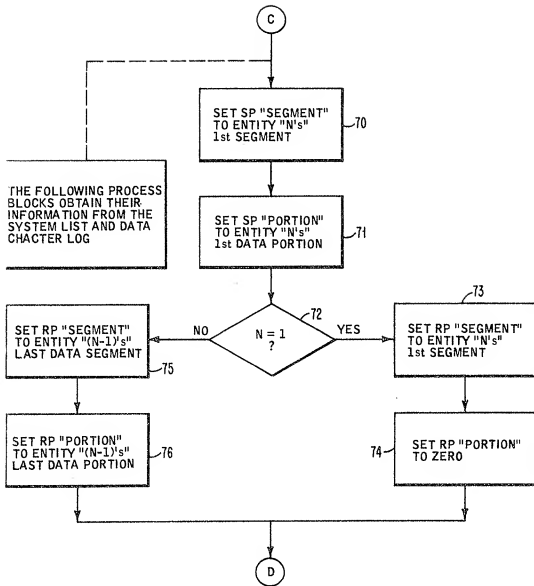


FIG. 7

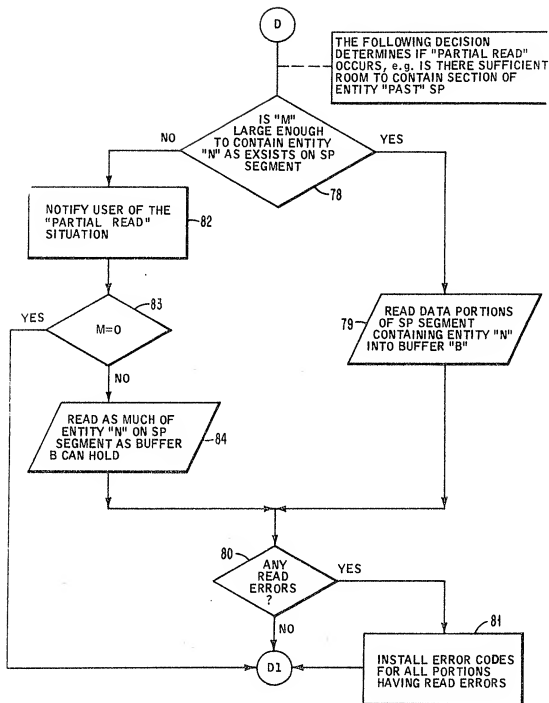


FIG. 8

*Clifford Montgomery, Renault, Clark, Kinsgarn, Barron & Howard*

Patent Agents

19-8

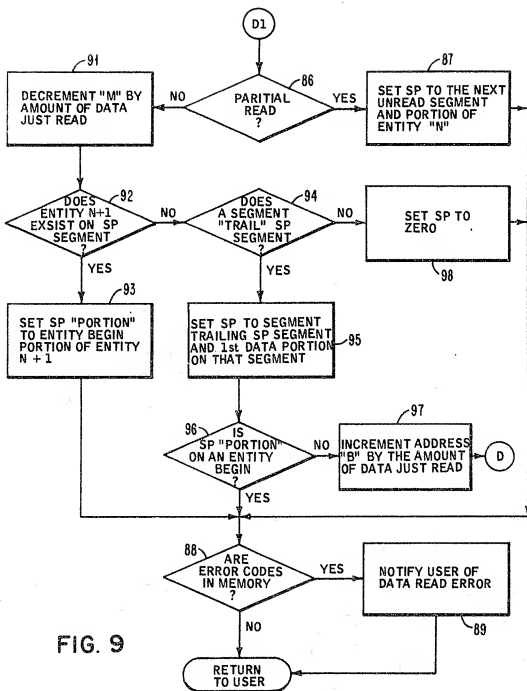
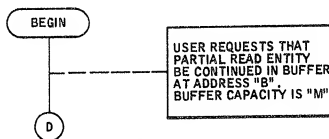
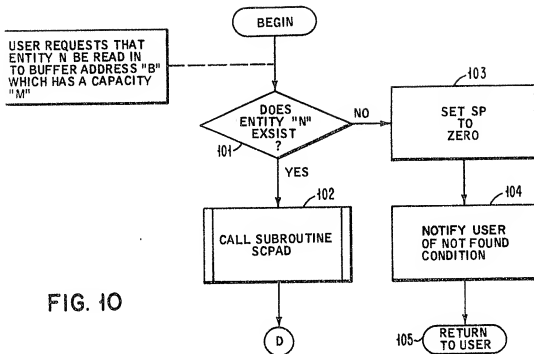
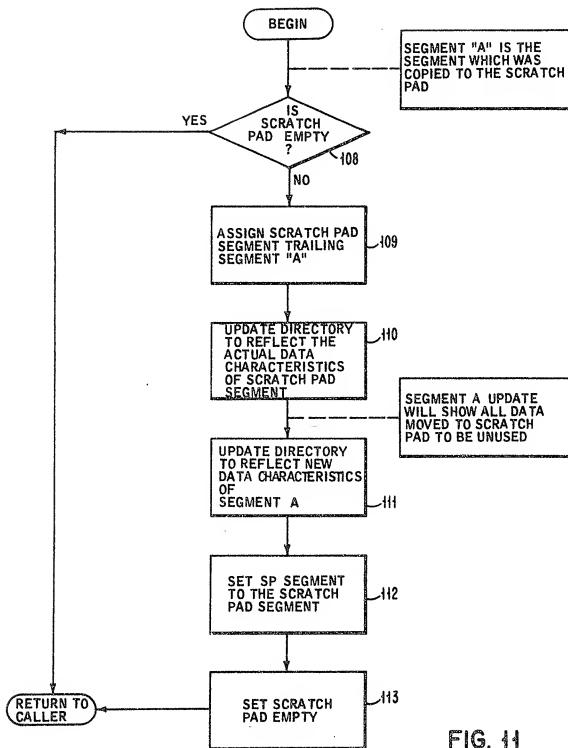


FIG. 9



19-10





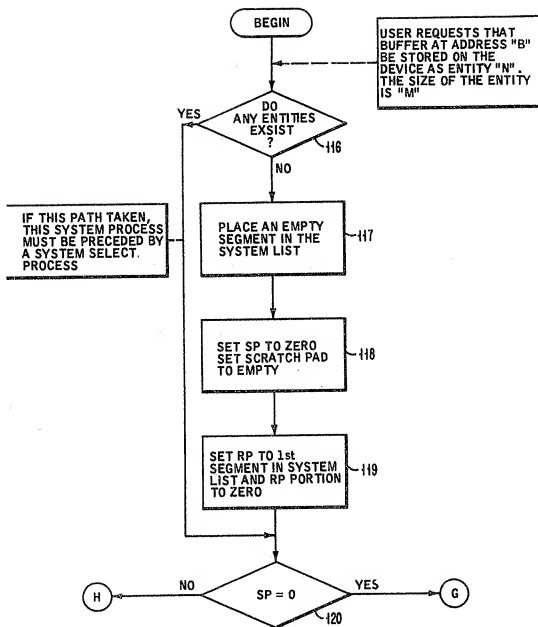


FIG. 13

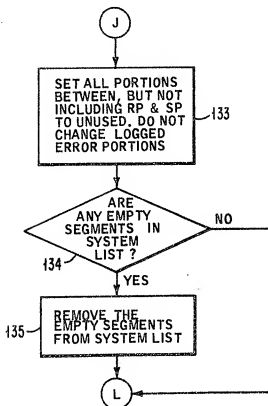
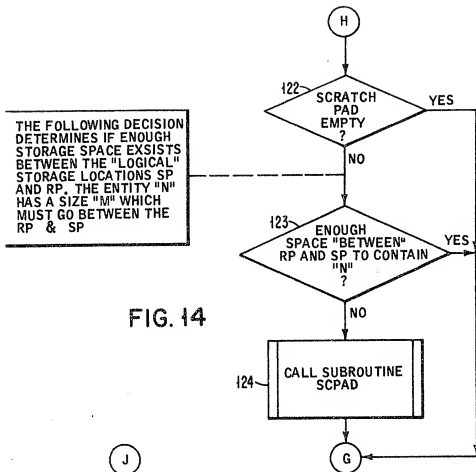
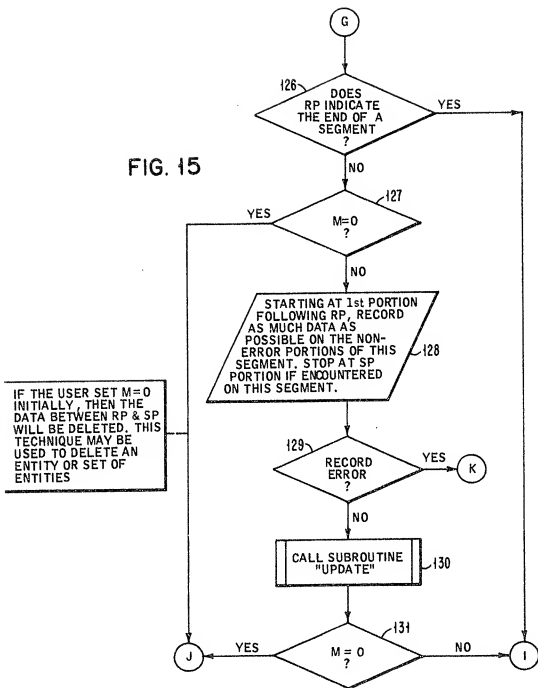


FIG. 15



19-14

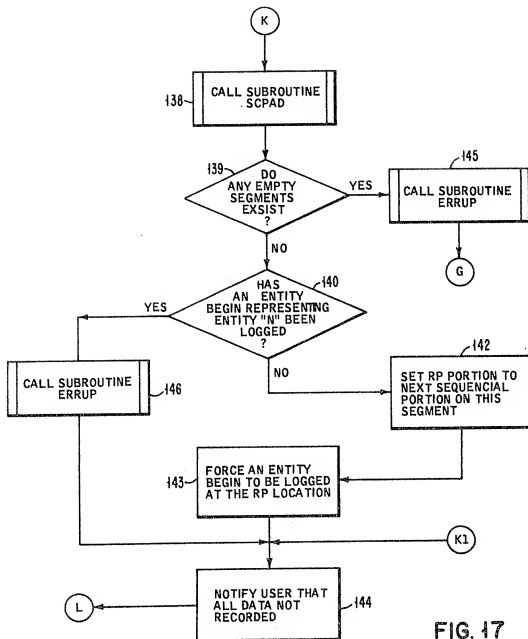


FIG. 17

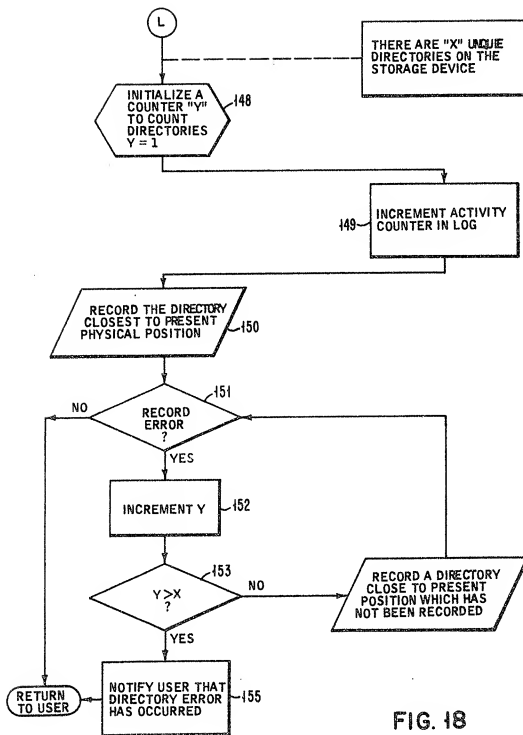


FIG. 18

19-16

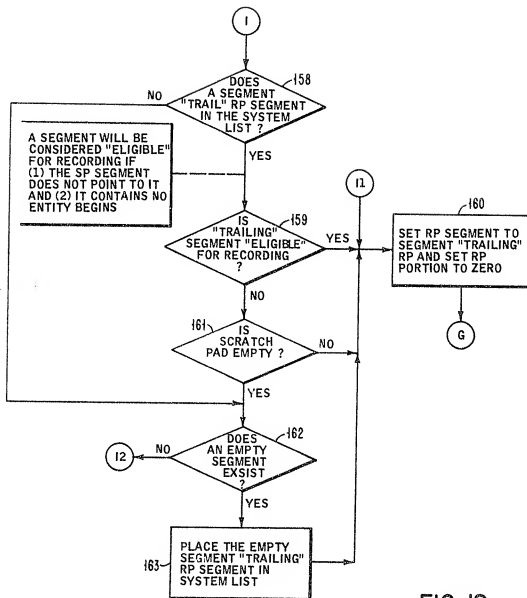


FIG. 19

19-17

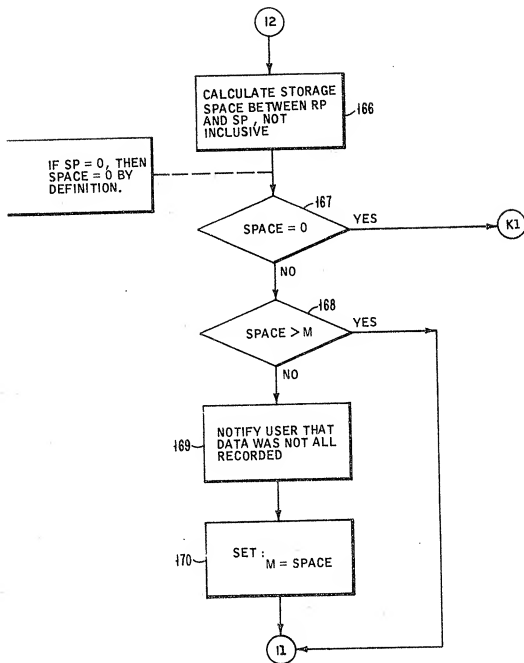


FIG. 20

Clifton, Irving, Bennett, Clark, Knapik, Harwood & Harwood

Patent Agents

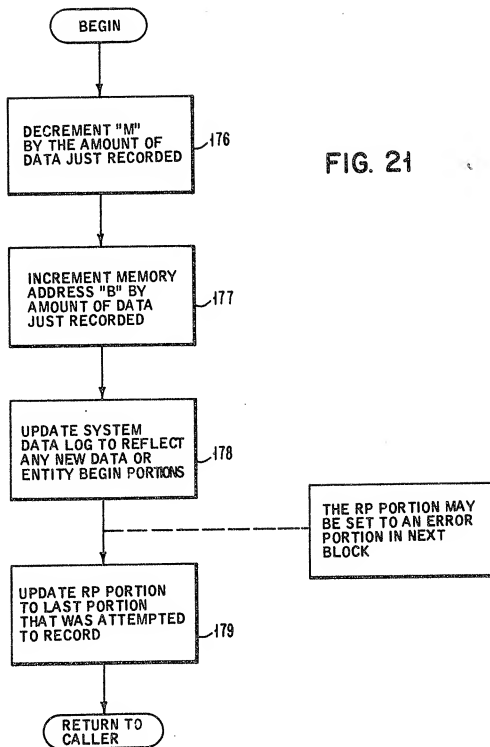




FIG. 22

